Document Audience:	SPECTRUM
Document ID:	229567
Old Document ID:	(formerly 72033)
Title:	Correct Use of the Driver Configuration File to Force the Sun[TM] GigaSwift Ethernet Adapter's Operational Parameters
Copyright Notice:	
Update Date:	Thu Oct 23 00:00:00 MDT 2008

Solution Type **Technical Instruction**

Solution 229567 : Correct Use of the Driver Configuration File to Force the Sun[TM] GigaSwift Ethernet Adapter's Operational Parameters

Related Categories

- Home>Product>Software>Network

Description

The recommendation is to implement auto-negotiation in fast and gigabit Ethernet environments

This is discussed in < Solution: 208373 > and you are encouraged to read this before continuing any further.

However, circumstances may arise where it is necessary or desirable to force the operational parameters on the interface.

The purpose of this document is to show how to use the GigaSwift NIC driver configuration file in order to do this.

Steps to Follow

Correct Use of the Driver Configuration File to Force the Sun[TM] GigaSwift Ethernet Adapter's

1. Introduction

GigaSwift is the third generation of Sun's gigabit Ethernet products and is supported by the ce(7D) STREAMS hardware driver. It is capable of 1000 Mbit/s (or 1 Gbit/s) maximum speed over both copper and fibre media and provides support for VLANs.

Note: The GigaSwift NIC will auto-negotiate by default and you should not alter the operational parameters if this is the desired mode of operation. For correct operation of the NIC you should allow both ends of the link to auto-negotiate or force both ends to use the same settings.

While the ndd command may be used to achieve the same results as the driver configuration file its use is less rigorous leading to greater scope for error. Furthermore, the use of ndd to make static changes is not recommended by Sun Microsystems. By "static" we mean changes which will remain in place even if the system is rebooted.

Note: The manual page for ndd(1M) states the following.

"The parameters supported by each driver may change from release to release. Like programs that read /dev/kmem, user programs or shell scripts that execute ndd should be prepared for parameter names to change."

In the past it has also been possible to use the /etc/system file to configure NIC parameters but this cannot be done with GigaSwift.

You should be aware of the following Sun Alert which is dated 15th October 2003 and references bug report 4802741.

Sun Alert ID: 55360
Synopsis: Sun Systems With Sun GigaSwift Ethernet 1.0 UTP Adapters (RJ-45)

```
May Not Link Up in "Forced" Mode For 1000 Mbps.
```

If, after forcing operational parameters on your GigaSwift interface(s), messages similar to the following appear on the console you should consult this document.

```
genunix: WARNING: ce1: fault detected external to device; service degraded
genunix: WARNING: ce1: xcvr addr:0x01 - link down
genunix: WARNING: ce0: fault detected external to device; service degraded
genunix: WARNING: ce0: xcvr addr:0x01 - link down
```

2. Using the Configuration File

2.1 Where to Place the File

General information on driver configuration files may be found in the manual page for driver.conf(4).

The GigaSwift driver configuration file is named "ce.conf" and you may see documentation which suggests that this file should be placed in the /platform/sun4u/kernel/drv directory. In fact it may be placed in any of the directories listed in the kernel's module_path variable which defines the search path for loadable kernel modules.

In Solaris 2.6 and 7 you can use adb(1) to view this path, e.g.,

```
# echo '*module_path/s' | adb -k
```

physmem 1e07
__tnf_probe_list_head+4: /platform/SUNW,Ultra-1/kernel
/platform/sun4u/kernel /kernel /usr/kernel

In versions of Solaris later than 7 adb has been replaced by mdb(1), e.g.,

```
# echo '*module_path/s' | mdb -k
0x10429228:     /platform/SUNW,Sun-Fire-480R/kernel /platform/sun4u/kernel
/kernel /usr/kernel
```

So, in this latter case, we could put the file in any of the directories listed below.

```
/platform/SUNW,Sun-Fire-480R/kernel/drv
/platform/sun4u/kernel/drv
/kernel/drv
/usr/kernel/drv
```

However, for ease of administration it is advisable to put the configuration file in the same directory as the driver itself.

The 64-bit kernel was introduced in the Solaris 7 operating system and you can use the isainfo(1) command to determine the number of bits in the address space of the native instruction set, i.e.,

```
# isainfo -b
64
```

On systems capable of supporting 64-bit drivers the configuration file should be placed in the directory in which the 32-bit driver would be located. For example, the configuration file for a 64-bit driver stored in /usr/kernel/drv/sparcv9 should be placed in the /usr/kernel/drv directory.

In order to clarify this, consider the following example taken from a system running Solaris 9.

```
# for d in `echo '*module_path/s' | mdb -k | tr ':' ' ' | tr -d "\n"`
> do
> find $d -name "ce" -print 2>/dev/null
> done
/platform/sun4u/kernel/drv/sparcv9/ce
```

```
/platform/sun4u/kernel/drv/ce

# file /platform/sun4u/kernel/drv/sparcv9/ce
/platform/sun4u/kernel/drv/sparcv9/ce:  ELF 64-bit MSB relocatable SPARCV9
Version 1

# file /platform/sun4u/kernel/drv/ce
/platform/sun4u/kernel/drv/ce:  ELF 32-bit MSB relocatable SPARC32PLUS
Version 1, V8+ Required
```

We see that there is a 64-bit driver located in the /platform/sun4u/kernel/drv/sparcv9 directory and a 32-bit driver in the /platform/sun4u/kernel/drv directory.

In this case the configuration file should be located in /platform/sun4u/kernel/drv even if we are running the 64-bit operating system.

2.2 Example File Content

You will need to determine the instance numbers of the ce interfaces on your system. This information is contained in the /etc/path_to_inst file which is maintained by the Solaris operating system and should not be edited without first consulting Sun Services.

Example 1

As a first example let's look at the devices on a Sun Fire 480R. If we look at the path_to_inst file on this type of system we may see something similar to the following.

```
# grep '"ce"' /etc/path_to_inst
"/pci@9,700000/network@2" 0 "ce"
"/pci@9,600000/network@1" 1 "ce"
```

Each line in the file contains the following three fields as described in the man page for path_to_inst(4).

"physical-name" instance-number "driver-binding-name"

So, we see two instances of the device, 0 and 1, which will be seen as ce0 and ce1 respectively in the output from the ifconfig -a command if the interfaces are plumbed.

To force ce0 to run at 100 Mbit/s full duplex and ce1 to run at 1 Gbit/s full duplex we would need to place the following two lines in the ce.conf file.

```
name="ce" parent="/pci@9,700000" unit-address="2" adv_autoneg_cap=0 adv_1000fdx_cap=0
adv_1000hdx_cap=0 adv_100fdx_cap=1 adv_100hdx_cap=0 adv_10fdx_cap=0 adv_10hdx_cap=0;

name="ce" parent="/pci@9,600000" unit-address="1" adv_autoneg_cap=0 adv_1000fdx_cap=1
adv_1000hdx_cap=0 adv_100fdx_cap=0 adv_100hdx_cap=0 adv_10fdx_cap=0 adv_10hdx_cap=0;
```

The important points to note here are:

1. The name, parent and unit-address parameters should be enclosed in double quotes. Should you wish to force all ce interfaces to the same speed and duplex you do not need to include these parameters.
   - The "name" parameter is the driver binding name for the device and will always be "ce". You may have seen previous revisions of this document which state that it should be "pci108e,abba". While this works you should use the driver binding name (ce) due to its simplicity and the fact that it does not tie the configuration to a specific type of PCI card as defined in the /etc/driver_aliases file.
   - The "parent" parameter is that part of the device path (or physical-name) prior to the "/network@" part. In a Sun Cluster environment you may see the device path prefixed with "node@<*n*>" where <*n*> is replaced by a digit. This prefix should **not** be included in the parent parameter.
   - The "unit-address" parameter is the number immediately after the "network@" part in the device path, it is not the device instance.

2. We have defined more than the minimum set of parameters to effect this change since we are being deliberately verbose for the sake of completeness and to reduce the scope for error.
3. Each entry in the file should be on a separate line and terminated with a semi-colon. While the example above may at first sight appear to be spread over several lines there are in fact only two lines, one for each instance of the device.
4. These parameters will take effect following a system reboot, only the ndd command may be used to effect dynamic changes.

Example 2

As a second example let's look at the interfaces on a Sun Fire 4800 domain where the /etc/path_to_inst file contains the following entries.

```
"/ssm@0,0/pci@18,600000/pci@1/network@0" 0 "ce"
"/ssm@0,0/pci@18,600000/pci@1/network@1" 1 "ce"
"/ssm@0,0/pci@19,700000/pci@3/network@0" 2 "ce"
"/ssm@0,0/pci@1c,600000/pci@1/network@0" 3 "ce"
"/ssm@0,0/pci@1c,600000/pci@1/network@1" 4 "ce"
"/ssm@0,0/pci@1d,700000/network@3" 5 "ce"
```

In order to force ce5 to 1000 Mbit/s full duplex and ce0 and ce3 to 100Mbit/s full duplex the ce.conf file should contain the following three lines.

```
name="ce" parent="/ssm@0,0/pci@1d,700000" unit-address="3" adv_autoneg_cap=0 adv_1000fdx_cap=1
adv_1000hdx_cap=0 adv_100fdx_cap=0  adv_100hdx_cap=0 adv_10fdx_cap=0 adv_10hdx_cap=0;

name="ce" parent="/ssm@0,0/pci@18,600000/pci@1" unit-address="0" adv_autoneg_cap=0
adv_1000fdx_cap=0 adv_1000hdx_cap=0 adv_100fdx_cap=1 adv_100hdx_cap=0 adv_10fdx_cap=0 adv_10h

name="ce" parent="/ssm@0,0/pci@1c,600000/pci@1" unit-address="0" adv_autoneg_cap=0
adv_1000fdx_cap=0 adv_1000hdx_cap=0 adv_100fdx_cap=1 adv_100hdx_cap=0 adv_10fdx_cap=0 adv_10h
```

In order to clarify point 1 above, if we wanted to force all six interfaces to operate at 1Gbit/s full duplex we would only need to place the following single line in the file.

```
adv_autoneg_cap=0 adv_1000fdx_cap=1 adv_1000hdx_cap=0 adv_100fdx_cap=0 adv_100hdx_cap=0 adv_1
adv_10hdx_cap=0;
```

3. Checking the Interface Settings

3.1 Relevant Parameters

Once the system has rebooted you will want to check that the interfaces are operating as expected.

The relevant parameters may be broken down into four distinct sections;

1. NIC capabilities.
   The names of these parameters begin with "cap_" and they show the capabilities which the NIC posesses. Note that these may, or may not, be enabled on the link.
2. Advertised capabilities.
   The names of these parameters begin "adv_" and they show the capabilities which the NIC is currently advertising on the link.
3. Actual mode.
   The names of these parameters begin "link_" and they show the mode in which the NIC is currently operating.
4. Link partner capabilities.
   The names of these parameters begin "lp_cap_" and they show the capabilities of the link partner as currently seen by the interface.

With the exception of the advertised capabilites these parameters must be displayed using the kstat(1M) command in versions of Solaris later than 7. For earlier versions you will need to use the undocumented netstat -k command, e.g.,

```
# netstat -k ce0
```

```
# netstat -k ce1
```

The advertised capabilities are displayed using the ndd(1M) command as shown in section 3.3 below.

3.2 NIC Capabilities

The important parameters and how to interpret them is shown in table 1 below.

Parameter Name	Value	Meaning
cap_autoneg	0	The NIC is not capable of auto-negotiation
	1	The NIC is capable of auto-negotiation
cap_1000fdx	0	The NIC is not capable of operating at 1000 Mbit/s full duplex
	1	The NIC is capable of operating at 1000 Mbit/s full duplex
cap_1000hdx	0	The NIC is not capable of operating at 1000 Mbit/s half duplex
	1	The NIC is capable of operating at 1000 Mbit/s half duplex
cap_100fdx	0	The NIC is not capable of operating at 100 Mbit/s full duplex
	1	The NIC is capable of operating at 100 Mbit/s full duplex
cap_100hdx	0	The NIC is not capable of operating at 100 Mbit/s half duplex
	1	The NIC is capable of operating at 100 Mbit/s half duplex
cap_10fdx	0	The NIC is not capable of operating at 10 Mbit/s full duplex
	1	The NIC is capable of operating at 10 Mbit/s full duplex
cap_10hdx	0	The NIC is not capable of operating at 10 Mbit/s half duplex
	1	The NIC is capable of operating at 10 Mbit/s half duplex

Table 1 - Interpretation of Parameters Showing the NIC Capabilities

As an example, let's look at the interface capabilities on our 480R.

```
# kstat -p ce:0::"/^cap_/"
ce:0:ce0:cap_1000fdx    1
ce:0:ce0:cap_1000hdx    1
ce:0:ce0:cap_100T4      0
ce:0:ce0:cap_100fdx     1
ce:0:ce0:cap_100hdx     1
ce:0:ce0:cap_10fdx      1
ce:0:ce0:cap_10hdx      1
ce:0:ce0:cap_asmpause   0
ce:0:ce0:cap_autoneg    1
ce:0:ce0:cap_pause      0

# kstat -p ce:1::"/^cap_/"
ce:1:ce1:cap_1000fdx    1
ce:1:ce1:cap_1000hdx    1
ce:1:ce1:cap_100T4      0
ce:1:ce1:cap_100fdx     1
ce:1:ce1:cap_100hdx     1
ce:1:ce1:cap_10fdx      1
ce:1:ce1:cap_10hdx      1
ce:1:ce1:cap_asmpause   0
ce:1:ce1:cap_autoneg    1
ce:1:ce1:cap_pause      0
```

This shows that both ce0 and ce1 are capable of auto-negotiation (cap_autoneg = 1) and can operate at 10/100/1000 Mbit/s full or half duplex.

Note: Information contained within this document is specific to the GigaSwift family of Ethernet products and not all NICs

which use the ce(7D) STREAMS hardware driver are capable of operating at 1000 Mbit/s (or 1 Gbit/s).

3.3 Advertised Capabilities

The important parameters and how to interpret them is shown in table 2 below.

Parameter Name	Value	Meaning
adv_autoneg_cap	0	The interface has been forced directly into a particular operating mode without regard to auto-negotiation
	1	The interface is currently auto-negotiating
adv_1000fdx_cap	0	The interface is not capable of operating at 1000 Mbit/s full duplex
	1	The interface is capable of operating at 1000 Mbit/s full duplex
adv_1000hdx_cap	0	The interface is not capable of operating at 1000 Mbit/s half duplex
	1	The interface is capable of operating at 1000 Mbit/s half duplex
adv_100fdx_cap	0	The interface is not capable of operating at 100 Mbit/s full duplex
	1	The interface is capable of operating at 100 Mbit/s full duplex
adv_100hdx_cap	0	The interface is not capable of operating at 100 Mbit/s half duplex
	1	The interface is capable of operating at 100 Mbit/s half duplex
adv_10fdx_cap	0	The interface is not capable of operating at 10 Mbit/s full duplex
	1	The interface is capable of operating at 10 Mbit/s full duplex
adv_10hdx_cap	0	The interface is not capable of operating at 10 Mbit/s half duplex
	1	The interface is capable of operating at 10 Mbit/s half duplex

Table 2 - Interpretation of Parameters Showing the Advertised Capabilities

We can check whether the NIC is actually auto-negotiating by using the ndd command. It is important to realise that when using this command you must first define which instance of the device you wish to query, e.g.,

```
# ndd -set /dev/ce instance 0
# ndd -get /dev/ce adv_autoneg_cap
1
```

This result shows that the ce interface with instance number "0" (ce0) is currently auto-negotiating. You may wish to take this a step further and see what capabilities it is advertising to the link partner.

To see a full list of the parameters which you can query you should run the command,

```
# ndd -get /dev/ce \?
```

The advertised capabilities are held in parameters with names beginning "adv_", e.g.,

```
# ndd -get /dev/ce \? | grep "adv_"
adv_autoneg_cap               (read and write)
adv_1000fdx_cap               (read and write)
adv_1000hdx_cap               (read and write)
adv_100T4_cap                 (read and write)
adv_100fdx_cap                (read and write)
adv_100hdx_cap                (read and write)
adv_10fdx_cap                 (read and write)
adv_10hdx_cap                 (read and write)
adv_asmpause_cap              (read and write)
adv_pause_cap                 (read and write)
```

So, we can run the following.

```
# ndd -set /dev/ce instance 0
# ndd -get /dev/ce adv_100fdx_cap
1
```

This shows that ce0 is advertising the fact that it can run at 100Mbit/s full duplex.

3.4 Actual Mode

Since we now know that ce0 is auto-negotiating, let's see what mode it is actually operating in at present.

```
# kstat -p ce:0::"/^link_/"
ce:0:ce0:link_T4        0
ce:0:ce0:link_asmpause  0
ce:0:ce0:link_duplex    1
ce:0:ce0:link_pause     0
ce:0:ce0:link_speed     100
ce:0:ce0:link_up        1

# kstat -p ce:1::"/^link_/"
ce:1:ce1:link_T4        0
ce:1:ce1:link_asmpause  0
ce:1:ce1:link_duplex    0
ce:1:ce1:link_pause     0
ce:1:ce1:link_speed     0
ce:1:ce1:link_up        0
```

The important parameters are interpreted as shown in table 3 below.

Parameter Name	Value	Meaning
link_up	0	The link is down
	1	The link is up
link_speed	1000	The link is operating at 1000 Mbit/s
	100	The link is operating at 100 Mbit/s
	10	The link is operating at 10 Mbit/s
link_duplex	0	The link is down
	1	The link is operating in half duplex mode
	2	The link is operating in full duplex mode

Table 3 - Interpretation of Parameters Showing the Actual Link Mode

So, in the example output shown above we see that ce1 is down (link_up=0) and that ce0 is up and running at 100 Mbit/s half duplex.

3.5 Link Partner Capabilities

Finally, we can check the link partner capabilities, e.g.,

```
# kstat -p ce:0::"/^lp_cap_/"
ce:0:ce0:lp_cap_1000fdx 0
ce:0:ce0:lp_cap_1000hdx 0
ce:0:ce0:lp_cap_100T4   0
ce:0:ce0:lp_cap_100fdx  0
ce:0:ce0:lp_cap_100hdx  1
ce:0:ce0:lp_cap_10fdx   0
ce:0:ce0:lp_cap_10hdx   1
ce:0:ce0:lp_cap_asmpause        0
ce:0:ce0:lp_cap_autoneg 1
```

```
ce:0:ce0:lp_cap_pause   0

# kstat -p ce:1::"/^lp_cap_/"
ce:1:ce1:lp_cap_1000fdx 0
ce:1:ce1:lp_cap_1000hdx 0
ce:1:ce1:lp_cap_100T4   0
ce:1:ce1:lp_cap_100fdx  0
ce:1:ce1:lp_cap_100hdx  0
ce:1:ce1:lp_cap_10fdx   0
ce:1:ce1:lp_cap_10hdx   0
ce:1:ce1:lp_cap_asmpause        0
ce:1:ce1:lp_cap_autoneg 0
ce:1:ce1:lp_cap_pause   0
```

The important parameters are interpreted as shown below in table 4.

Parameter Name	Vlaue	Meaning< /b>
lp_cap_autoneg	0	The link partner operational parameters have been forced or the link partner simply does not support auto-negotiation
	1	Auto-negotiation information has been received from the link partner
lp_cap_1000fdx	0	The link partner appears to be incapable of operating at 1000 Mbit/s full duplex
	1	The link partner is capable of operating at 1000 Mbit/s full duplex
lp_cap_1000hdx	0	The link partner appears to be incapable of operating at 1000 Mbit/s half duplex
	1	The link partner is capable of operating at 1000 Mbit/s half duplex
lp_cap_100fdx	0	The link partner appears to be incapable of operating at 100 Mbit/s full duplex
	1	The link partner is capable of operating at 100 Mbit/s full duplex
lp_cap_100hdx	0	The link partner appears to be incapable of operating at 100 Mbit/s half duplex
	1	The link partner is capable of operating at 100 Mbit/s half duplex
lp_cap_10fdx	0	The link partner appears to be incapable of operating at 10 Mbit/s full duplex
	1	The link partner is capable of operating at 10 Mbit/s full duplex
lp_cap_10hdx	0	The link partner appears to be incapable of operating at 10 Mbit/s half duplex
	1	The link partner is capable of operating at 10 Mbit/s half duplex

Table 4 - Interpretation of Parameters Showing the Link Partner Capabilities

Looking at ce0 we see that the link partner is auto-negotiating (lp_cap_autoneg=1) and is capable of operating at 100 Mbit/s half duplex and 10 Mbit/s half duplex.

Interpretation of these values when the link is not up (link_up=0) is beyond the scope of a knowledge article and is likely to change between revisions of the driver.

Product

```
Sun Gigabit Ethernet PCI Adapter
```

Keywords

```
Cassini, ce, ce.conf, Ethernet, GigaSwift, gigabit, forced, duplex, speed, auto-negotiation, r
```

Previously Published As

72033

Attachments

This solution has no attachment